

17/409
4/3/02

PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Toshihiko Nakamura	Examiner:	Unassigned
Serial No:	Unassigned	Art Unit:	Unassigned
Filed:	Herewith	Docket:	14612
For:	APPARATUS AND METHOD FOR PRODUCING PERFORMANCE EVALUATION MODEL		Dated: May 11, 2001

1c903 U.S. PRO
09/054159
05/11/01

Assistant Commissioner for Patents
Washington, D.C. 20231

CLAIM OF PRIORITY

Sir:

Applicant in the above-identified application hereby claims the right of priority in connection with Title 35 U.S.C. § 119 and in support thereof, herewith submits a certified copy of Japanese Patent Application No.2000-138391, filed on May 11, 2000.

Respectfully submitted,

Paul J. Esatto, Jr.
Paul J. Esatto, Jr.
Registration No.: 30,749

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

Express Mailing Label No.: EL 823707655 US

Date of Deposit: May 11, 2001

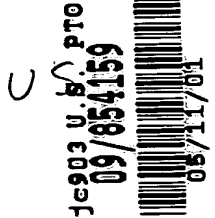
I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. § 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents and Trademarks, Washington, D.C. 20231 on May 11, 2001.

Dated: May 11, 2001

Janet Grossman
Janet Grossman

05P1088/

日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: 2000年 5月11日

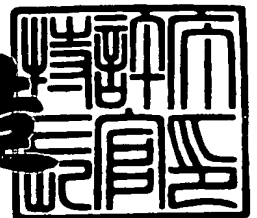
出願番号
Application Number: 特願2000-138391

出願人
Applicant(s): 日本電気株式会社

2001年 3月 9日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2001-3017348

【書類名】 特許願

【整理番号】 37300356

【あて先】 特許庁長官殿

【国際特許分類】 G06F 11/28

【発明者】

 【住所又は居所】 東京都港区芝五丁目 7 番 1 号 日本電気株式会社内

 【氏名】 中村 寿彦

【特許出願人】

 【識別番号】 000004237

 【氏名又は名称】 日本電気株式会社

【代理人】

 【識別番号】 100088890

 【弁理士】

 【氏名又は名称】 河原 純一

【手数料の表示】

 【予納台帳番号】 009690

 【納付金額】 21,000円

【提出物件の目録】

 【物件名】 明細書 1

 【物件名】 図面 1

 【物件名】 要約書 1

 【包括委任状番号】 9001717

【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 性能評価モデル生成装置および性能評価モデル生成方法

【特許請求の範囲】

【請求項 1】 変換規則を格納する変換規則格納部と、
UMLモデルを入力して解析するUMLモデル解析部と、
前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部と
を有することを特徴とする性能評価モデル生成装置。

【請求項 2】 利用者からの変換規則を入力する変換規則編集部と、
前記変換規則編集部により入力された変換規則を格納する変換規則格納部と、
UMLモデルを入力して解析するUMLモデル解析部と、
前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部と
を有することを特徴とする性能評価モデル生成装置。

【請求項 3】 UMLモデルの拡張記法を使い、ハードウェアリソースに割り付ける表記を定め、その表記に従ったUMLモデルから性能評価モデルへの変換規則を決め、この変換規則によってUMLモデルから系統的に性能評価モデルを作成することを特徴とする性能評価モデル生成方法。

【請求項 4】 シーケンス図の最初のメッセージをカレントメッセージにセットする工程と、
シーケンス図のカレントメッセージに着目する工程と、
送信先オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、
送信元オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、
送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに基づいて変換規則を検索する工程と、
検索された変換規則に従ってカレントメッセージを性能評価ツールのノードに変換し配置する工程と、
カレントメッセージの次のメッセージがあるかどうかを判定する工程と、
次のメッセージがあればこれをカレントメッセージとして制御を前記カレントメ

ッセージに着目する工程に戻す工程と

を含むことを特徴とする性能評価モデル生成方法。

【請求項 5】前記クラスタイプおよび属性を調査する工程が、さらに、オブジェクトに着目する工程と、オブジェクトを定義するクラスに着目する工程と、オブジェクトから対応するクラスを探し該クラスに繋がるリソース関連線を探す工程と、リソース関連線が存在するかどうかを判定する工程と、リソース関連線が存在すればリソース関連のあるリソースクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程と、リソース関連線が存在しなければ元のクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程とからなる請求項 4 記載の性能評価モデル生成方法。

【請求項 6】コンピュータを、変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録した記録媒体。

【請求項 7】コンピュータを、利用者からの変換規則を入力する変換規則編集部、前記変換規則編集部により入力された変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録した記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は性能評価モデル生成装置および性能評価モデル生成方法に関し、特にUML(Unified Modeling Language)モデルから性能評価モデルを生成する性能評価モデル生成装置および性能評価モデル生成方法に関する。

【0002】

【従来の技術】

製品サイクルの短期化のため、サービス分析から実現までの時間的短縮が望まれている。また、組み込み機器分野においては、情報端末の普及によって、より多機能かつ高機能なシステムが求められ、また、システム L S I (L a r g e S c a l e d I n t e g r a t i o n) の登場によって、ハードウェアとソフトウェアとをより密にシステム設計する必要が生じている。

【0003】

このような状況の下で、システム構成を決めて実装した後に評価を行っているようでは、時間的な短縮は望めない。そこで、システム構成を決定するに当たって、実装することなく性能などの見積もりを行う必要性が生じてくる。このニーズに対して、待ち行列理論を使った性能評価ツールが使われている。このような性能評価ツールでは、システムに入力されるジョブを待ち行列を使ってプロセッサに割り当てるという性能評価モデルを作成し、処理速度などのパラメータを与えることで性能評価を行っている。この性能評価モデルを使った性能評価は、性能評価モデルを作成するコストが高いため、強力な性能評価ができるにもかかわらず、広く使われるに至っていない。

【0004】

特開平 7 - 8 4 8 3 1 号公報では、このような性能評価の支援手法として、過去に性能評価したモデル（性能評価モデル）をデータベースに保持し、以降の設計モデルの作成に役立てるようなツールが提案されている。このツールでは、性能評価をある一定期間続けることでツールに依存することなく性能評価モデルが蓄積されて、設計モデルの作成が容易になるというメリットが得られる。

【0005】

【発明が解決しようとする課題】

しかし、上述した従来のツールによっても、システムをゼロから開発する場合で、性能評価モデルに類似のものが無いときには、性能評価モデルをゼロから作成する必要があるため、性能評価モデルを作成するコストが高いという問題点に関してはなんら解決されていなかった。

【0006】

本発明の目的は、モデル作成のための事実上の標準言語であるUMLで記述されたUMLモデルから性能評価モデルを自動的に生成する性能評価モデル生成装置を提供することにある。

【0007】

また、本発明の他の目的は、UMLで記述されたUMLモデルから性能評価モデルを自動的に生成する性能評価モデル生成方法を提供することにある。

【0008】

【課題を解決するための手段】

本発明の性能評価モデル生成装置は、変換規則を格納する変換規則格納部と、UMLモデルを入力して解析するUMLモデル解析部と、前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有することを特徴とする。

【0009】

また、本発明の性能評価モデル生成装置は、利用者からの変換規則を入力する変換規則編集部と、前記変換規則編集部により入力された変換規則を格納する変換規則格納部と、UMLモデルを入力して解析するUMLモデル解析部と、前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部とを有することを特徴とする。

【0010】

さらに、本発明の性能評価モデル生成方法は、UMLモデルの拡張記法を使い、ハードウェアリソースに割り付ける表記を定め、その表記に従ったUMLモデルから性能評価モデルへの変換規則を決め、この変換規則によってUMLモデルから系統的に性能評価モデルを作成することを特徴とする。

【0011】

さらにまた、本発明の性能評価モデル生成方法は、シーケンス図の最初のメッセージをカレントメッセージにセットする工程と、シーケンス図のカレントメッセージに着目する工程と、送信先オブジェクトを定義するクラスのタイプおよび属性を調査する工程と、送信元オブジェクトを定義するクラスのタイプおよび属性

を調査する工程と、送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに基づいて変換規則を検索する工程と、検索された変換規則に従ってカレントメッセージを性能評価ツールのノードに変換し配置する工程と、カレントメッセージの次のメッセージがあるかどうかを判定する工程と、次のメッセージがあればこれをカレントメッセージとして制御を前記カレントメッセージに着目する工程に戻す工程とを含むことを特徴とする。特に、前記クラスタイプおよび属性を調査する工程が、さらに、オブジェクトに着目する工程と、オブジェクトを定義するクラスに着目する工程と、オブジェクトから対応するクラスを探し該クラスに繋がるリソース関連線を探す工程と、リソース関連線が存在するかどうかを判定する工程と、リソース関連線が存在すればリソース関連のあるリソースクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程と、リソース関連線が存在しなければ元のクラスのタイプおよび属性をクラスのタイプおよび属性として保存する工程とからなることを特徴とする。

【 0 0 1 2 】

一方、本発明の記録媒体は、コンピュータを、変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録する。

【 0 0 1 3 】

また、本発明の記録媒体は、コンピュータを、利用者からの変換規則を入力する変換規則編集部、前記変換規則編集部により入力された変換規則を格納する変換規則格納部、UMLモデルを入力して解析するUMLモデル解析部、および前記UMLモデル解析部による解析結果を前記変換規則格納部に格納された変換規則に従って変換して性能評価モデルを生成する変換処理部として機能させるためのプログラムを記録する。

【 0 0 1 4 】

【発明の実施の形態】

以下、本発明の実施の形態について図面を参照して詳細に説明する。

【 0 0 1 5 】

(1) 第 1 の実施の形態

図 1 は、本発明の第 1 の実施の形態に係る性能評価モデル生成装置の構成を示すブロック図である。本実施の形態に係る性能評価モデル生成装置は、UML モデル 5 を入力して解析する UML モデル解析部 1 と、UML モデル解析部 1 による解析結果を変換規則格納部 4 に格納された変換規則に従って変換して性能評価モデル 6 を生成する変換処理部 2 と、利用者からの変換規則を入力する変換規則編集部 3 と、変換規則編集部 3 により入力された変換規則を格納する変換規則格納部 4 とから構成されている。なお、図 1 中、符号 7 は UML モデル 5 を生成する UML モデル入力ツール、8 は性能評価モデル 6 を性能評価する性能評価ツールをそれぞれ示す。

【 0 0 1 6 】

図 2 は、システムの動作を記述した UML モデル 5 から性能評価モデル 6 を系統的に生成する性能評価モデル生成方法を説明する図である。UML モデル 5 は、全て UML という統一モデリング言語の規格に従った記法で記述する。システムの動作は、動的な振る舞いを表すシーケンス図 5 1 と、システムの静的な特徴を表すクラス図 5 2 とで表現される。第 1 の実施の形態では、UML モデル 5 の記述単位として、UML モデル 5 をパッケージ 5 0 という単位に分割する。つまり、UML モデル 5 は 1 つ以上のパッケージ 5 0 から構成され、パッケージ 5 0 毎にシーケンス図 5 1 およびクラス図 5 2 を用意する。また、1 つのパッケージ 5 0 内のシーケンス図 5 1 およびクラス図 5 2 から生成される性能評価モデル 6 の単位を、性能評価サブモデル 6 0 と呼ぶ。つまり、性能評価モデル 6 は、1 つ以上の性能評価サブモデル 6 0 から構成されている。

【 0 0 1 7 】

図 3 を参照すると、シーケンス図 5 1 は、システムの動的な関係をモデル化した図である。シーケンス図 5 1 の構成要素の中で、性能評価サブモデル 6 0 の作成に関係するものには、オブジェクト O b 0 1 ～ O b 0 6 と、メッセージ M 0 1 ～ M 0 7 とがある。オブジェクト O b 0 1 ～ O b 0 6 は、機能単位に登場するクラスのインスタンスである。具体的には、後述する図 4 のクラス図 5 2 中に対応す

るクラスC 0 1～C 0 5のインスタンスである。メッセージM 0 1～M 0 7は、クラスC 0 1～C 0 5の実際の動作手順を時系列に沿って配置したものであり、上の方が相対的に早い時間を表している。破線矢印で示されたものは、手続き呼び出しの戻りを示している。戻りの破線矢印は省略してもよい。

【 0 0 1 8 】

図 4 は、クラス図 5 2 およびクラス図 5 2 に対して性能評価サブモデル 6 0 を作成するために行われる拡張表記を説明する図である。クラス図 5 2 は、システムを構成するクラスの静的な関連を示した図である。クラスは、モデリングに使用されるモデル要素であり、システムを構成する通常のオブジェクト指向設計で使われるクラス（以下、システムクラスという）と、ハードウェアリソースを表すクラス（以下、リソースクラスという）とに大きく区別され、それぞれさらに細かく分類される。この分類については、以下の[クラスの分類]で詳しく説明する。図 4 においては、クラスC 0 1～C 0 5 がシステムクラスである。性能評価のためにクラス図 5 2 に付加する拡張表記は、1 つが、処理に必要なハードウェアリソースを表すリソースクラスC 1 1, C 1 2, C 1 3として記述する。もう1 つは、これらのハードウェアリソースへの関連記述であり、これをリソース関連線L 0 1, L 0 2, L 0 3と名付ける。また、関連自体をリソース関連と名付け、システムクラスC 0 2, C 0 3, C 0 4をリソース関連線L 0 1, L 0 2, L 0 3を使ってリソースクラスC 1 1, C 1 2, C 1 3と結び付けることで、リソースクラスC 1 1, C 1 2, C 1 3のハードウェアリソースにシステムクラスC 0 2, C 0 3, C 0 4を割り付けるという意味づけをする。このリソース関連により、性能評価サブモデル 6 0 の生成を可能としている。

【 0 0 1 9 】

変換規則格納部 4 中の変換規則は、以下に示す[クラスの分類]と、[性能評価モデルの変換規則]とで構成されている。

【 0 0 2 0 】

[クラスの分類]

◎システムクラス

- ・タイプ<<Active>>のクラス

通常のクラス。属性およびメソッドを持ち、自立的に動作するクラス。

【0021】

・タイプ<<Data>>のクラス

属性が中心のクラス。メソッドは、所有する属性の読み書きのみに限定されている。

【0022】

・タイプ<<Interface>>のクラス

タイプ<<Interface>>のクラスは、パッケージ50内に唯一のクラスであり、外部パッケージからカレントパッケージへの全てのメッセージを受け取り、カレントパッケージ内の各クラスにメッセージを送出する。

【0023】

・タイプ<<Actor>>のクラス

タイプ<<Actor>>のクラスは、システムの外部に存在し、外部からシステム内に何らかの働きかけを行うクラス的一种である。アクターがシステム内部にも記憶域を持つ場合、クラス図52にタイプ<<Actor>>のクラスとタイプ<<Storage>>のクラスとのクラス関連を記述する関係上、アクターの記述が必要となる（後述する[変換規則]の a. アクターを参照）。これ以外の場合、クラス図52にアクターの記述がなくてもよい。

【0024】

◎リソースクラス

リソースクラスは、ハードウェアリソースを表すクラスである。リソースクラスには、性能評価サブモデル60に対応するノード群がある。ハードウェアリソースの占有時間など、変換したノードに設定すべきパラメータをクラスの属性に記述し、そのパラメータを使って性能評価サブモデル60のノードに変換する。性能評価の対象にする必要がない場合、評価対象外と記し、性能評価サブモデル60のノードは生成しない。リソースクラスが対応する性能評価サブモデル60のノード群の例を、図7に示す。

【0025】

・タイプ<<Storage>>のクラス

タイプ<<Storage>>のクラスは、ハードウェアリソースの中で記憶域を定義するクラスである。記憶域の種類や性質に応じて、タイプ<<Storage>>でさまざまなクラスを定義する。

【0026】

・タイプ<<Processing>>のクラス

タイプ<<Processing>>のクラスは、ハードウェアリソースの中で実際の処理をするハードウェアを定義するクラスである。ソフトウェア処理の場合のCPU (Central Processing Unit) も、このクラスで定義する。

【0027】

[性能評価モデルへの変換規則]

メッセージの送信元オブジェクトおよび送信先オブジェクトを定義するクラスのタイプに応じて、以下の変換規則を適用する。クラスのタイプを追加することで、この変換規則を追加することが可能となる。以下、「送信元オブジェクトを定義するクラスのタイプ → 送信先オブジェクトを定義するクラスのタイプ」という形式で変換規則を記述する。

【0028】

a. 「タイプ<<Actor>> → 任意のクラスタイプ」

タイプ<<Actor>>のクラスが最初のメッセージを送信する送信元オブジェクトになっている場合、データの発生源であるソースノードに変換する。

【0029】

b. 「外部パッケージの任意のクラスタイプ → カレントパッケージのタイプ<<Interface>>」

カレントパッケージのタイプ<<Interface>>のクラスに対するメッセージは、データの入り口であるエンターノードに変換する。

【0030】

c. 「カレントパッケージのタイプ<<Interface>> → 外部パッケージの任意のクラスタイプ」

カレントパッケージのタイプ<<Interface>>のクラスから外部パッ

ページへのメッセージは、データの出口であるリターンノードに変換する。

【0031】

d. 「任意のクラスタイプ → タイプ<<Processing>>」

タイプ<<Processing>>のクラスが定義するリソースノードを配置する。

【0032】

e. 「任意のクラスタイプ → タイプ<<Storage>>」

タイプ<<Storage>>のクラスが定義するリソースノードを配置する。

【0033】

f. 「任意のクラスタイプ → 外部パッケージのタイプ<<Interface>>」

外部パッケージのタイプ<<Interface>>のクラスがメッセージを受け取る場合、外部パッケージに対する処理の依頼であるので、その外部パッケージに対応する性能評価サブモデル60に変換する。これに続くメッセージが、カレントメッセージと逆方向になっている場合、これに続くメッセージを無視する。

【0034】

g. 破線矢印

破線矢印のメッセージは、それ自体省略しても可能なものである。よって、このメッセージは、変換の際に無視することができる。

【0035】

h. 終了処理

最後のメッセージでシンクノードまたはリターンノードが配置されない場合、このどちらかのノードを配置し、性能評価サブモデル60を完結させる。どちらのノードを配置するかは、変換した性能評価サブモデル60の最初のノードに着目し、シンクノードの場合はソースノード、エンターノードの場合はリターンノードを配置する。

【0036】

図5を参照すると、性能評価サブモデル60は、システム構成をノードで表現す

る。ノードは、大別すると、システムの処理フロー自体を表すノードと、処理フローで利用されるハードウェアリソースを表すノードとに区別できる。図 5 中の性能評価サブモデル 6 0 では、前者が下側に、後者が上側に記述されている。システムの処理フローを表現するノードは、そのフローをノード間の結線で表現している。このようなノードには、ハードウェアを消費する「処理」を表すサービスノード、データの入力を表すソースノード、処理の終了を表すシンクノードなどがある。一方、ハードウェアリソースを表すノードとして、ハードウェアに相当するリソースノードがある。リソースノードは、サービスノードでの処理によって消費される。この消費時間、優先順位などを動的パラメータ 6 1 として設定する。さらに、入力されるデータのパラメータを加えて、性能評価サブモデル 6 0 が生成できる。

【 0 0 3 7 】

図 5 は、性能評価サブモデル 6 0 の一例を記述する図である。なお、ここでは、性能評価ツール 8 として、待ち行列理論を使った S E S / W o r k b e n c h (S E S 社の登録商標) を想定している。この性能評価サブモデル 6 0 には、処理する側のノードとして、上部に、C P U を表すサービスノード N 0 1 が記述されている。一方、処理フローを表すノードとして、下部に、C P U を消費するサービスノード N 0 3 と、C P U とは無関係に時間を消費するサービスノード N 0 5 と、システムに入力されるデータに関連するソースノード N 0 2 およびシンクノード N 0 8 と、その他のノードとして、性能評価サブモデル 6 0 への参照を表すサブモデルノード N 0 7, リソース管理のアロケートノード N 0 4, リリースノード N 0 6 と、そして、その間の結線 3 1 1 ~ 3 1 6 が記述されている。最後に、上部と下部とを結び付ける記述として、動的パラメータ 6 1 が設定されている。変換では、リソース管理のアロケートノード N 0 4, メモリアクセスを表すサービスノード N 0 5, およびリリースノード N 0 6 という一連のノードを排他アクセス M e m o r y のモデルとしてあらかじめ用意しておき、これらをセットにして変換規則を適用する。図 5 にあるノード以外に、他の性能評価サブモデル 6 0 からデータの流れを引き継ぐエンターノードおよびリターンノードがある。

【 0 0 3 8 】

[性能評価サブモデルへの変換手順]

図 6 は、性能評価サブモデル 6 0 への変換手順を表すフローチャートである。性能評価サブモデル 6 0 は、あるパッケージ 5 0 に着目し、そのパッケージ 5 0 のシーケンス図 5 1 およびクラス図 5 2 から変換して生成される。この着目しているパッケージ 5 0 をカレントパッケージと呼び、それ以外のパッケージ 5 0 を外部パッケージと呼ぶ（図 4 参照）。

【 0 0 3 9 】

図 6 を参照すると、変換処理部 2 の処理は、カレントメッセージセットステップ S 1 0 1 と、カレントメッセージ着目ステップ S 1 0 2 と、送信元オブジェクトのクラスタイプおよび属性調査ステップ S 1 0 3 と、送信先オブジェクトのクラスタイプおよび属性調査ステップ S 1 0 4 と、変換規則検索ステップ S 1 0 5 と、ノード生成・連結ステップ S 1 0 6 と、次メッセージ有無判定ステップ S 1 0 7 と、オブジェクト着目ステップ S 1 0 8 と、クラス着目ステップ S 1 0 9 と、リソース関連探索ステップ S 1 1 0 と、リソース関連有無判定ステップ S 1 1 1 と、リソース関連先クラスタイプおよび属性保存ステップ S 1 1 2 と、自クラスタイプおよび属性保存ステップ S 1 1 3 とからなる。

【 0 0 4 0 】

次に、このように構成された第 1 の実施の形態に係る性能評価モデル生成装置の動作について、第 1 の実施の形態に係る性能評価モデル生成方法とともに説明する。

【 0 0 4 1 】

ここでは、図 3 のシーケンス図 5 1 および図 4 のクラス図 5 2 からメッセージを時系列に沿って図 5 の性能評価サブモデル 6 0 に変換する例について説明する。

【 0 0 4 2 】

利用者は、変換規則編集部 3 を使用して、あらかじめ変換規則を編集して変換規則格納部 4 に格納しておく。ここでは、既述した変換規則 a. ～ h. が格納されたものとする。

【 0 0 4 3 】

一方、UML モデル解析部 1 は、UML モデル入力ツール 7 を使って生成された

UMLモデル5を入力して解析する。詳しくは、性能評価サブモデル60に変換するために不必要な情報を削除し、必要な情報が揃っているかどうかをチェックし、必要な情報が揃っていたならば変換処理部2に解析結果を渡す。

【0044】

変換処理部2は、UMLモデル解析部1から渡された解析結果中の、シーケンス図51の最初のメッセージM01をカレントメッセージにセットする（ステップS101）。

【0045】

次に、変換処理部2は、シーケンス図51のカレントメッセージM01に着目する（ステップS102）。

【0046】

続いて、変換処理部2は、カレントメッセージM01の送信元オブジェクトOb01を定義するシステムクラスのタイプおよび属性を調査する（ステップS103）。

【0047】

詳しくは、変換処理部2は、まず、送信元オブジェクトOb01に着目し（ステップS108）、名前「actor」に基づいて送信元オブジェクトOb01を定義するクラス（ここでは、アクターはシステム内部に記憶域を持たないものとし、図4中には対応するクラスが図示されていない）に着目する（ステップS109）。次に、そのクラスに繋がるリソース関連線を探す（ステップS110）。リソース関連線が存在しないので（ステップS111）、元のクラスのタイプ<<Actor>>および属性をタイプおよび属性としてそのまま保存する（ステップS113）。

【0048】

次に、変換処理部2は、カレントメッセージM01の送信先オブジェクトOb02を定義するクラスC01のタイプおよび属性を調査する（ステップS104）。

【0049】

詳しくは、変換処理部2は、まず、送信先オブジェクトOb02に着目し（ステ

ップ S 1 0 8)、名前「MainP」に基づいて送信先オブジェクト Ob 0 2 を定義するシステムクラス C 0 1 に着目する(ステップ S 1 0 9)。次に、システムクラス C 0 1 に繋がるリソース関連線を探す(ステップ S 1 1 0)。リソース関連線が存在しないので(ステップ S 1 1 1)、元のシステムクラス C 0 1 のタイプ<<Interface>>および属性をシステムクラス C 0 1 のタイプおよび属性としてそのまま保存する(ステップ S 1 1 3)。

【0050】

続いて、変換処理部 2 は、送信元オブジェクト Ob 0 1 を定義するクラスのタイプ<<Actor>>および送信先オブジェクト Ob 0 2 を定義するシステムクラス C 0 1 のタイプ<<Interface>>に基づいて、変換規則格納部 4 から変換規則 a. を検索する(ステップ S 1 0 5)。

【0051】

続いて、変換処理部 2 は、検索された変換規則 a. 「アクターが最初のメッセージを送信するオブジェクトになっている場合、データの発生源であるソースノードに変換する。」に基づいて、ソースノード N 0 2 に変換して性能評価サブモデル 6 0 に配置する(ステップ S 1 0 6)。

【0052】

次に、変換処理部 2 は、シーケンス図 5 1 の次のメッセージの存在を調査し(ステップ S 1 0 7)、次のメッセージ M 0 2 があるので、ステップ S 1 0 2 に制御を戻す。

【0053】

変換処理部 2 は、次のメッセージ M 0 2 をカレントメッセージにセットして着目する(ステップ S 1 0 2)。

【0054】

次に、変換処理部 2 は、メッセージ M 0 2 の送信元オブジェクト Ob 0 2 を定義するシステムクラスのタイプおよび属性を調査する(ステップ S 1 0 3)。

【0055】

詳しくは、変換処理部 2 は、まず、送信元オブジェクト Ob 0 2 に着目し(ステップ S 1 0 8)、名前「MainP」に基づいて送信元オブジェクト Ob 0 2 を

定義するシステムクラスC01に着目する（ステップS109）。次に、システムクラスC01に繋がるリソース関連線を探す（ステップS110）。リソース関連線が存在しないので（ステップS111）、元のシステムクラスC01のタイプ<<Interface>>および属性をシステムクラスC01のタイプおよび属性としてそのまま保存する（ステップS113）。

【0056】

続いて、変換処理部2は、メッセージM02の送信先オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する（ステップS104）。

【0057】

詳しくは、変換処理部2は、まず、送信先オブジェクトOb03に着目し（ステップS108）、名前「Main」に基づいて送信先オブジェクトOb03を定義するシステムクラスC02に着目する（ステップS109）。次に、システムクラスC02に繋がるリソース関連線を探す（ステップS110）。リソース関連線L01が存在するので（ステップS111）、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプとして保存する（ステップS112）。

【0058】

次に、変換処理部2は、送信元オブジェクトOb02を定義するシステムクラスC01のタイプ<<Interface>>および送信先オブジェクトOb03を定義するシステムクラスC02のタイプ<<Processing>>に基づいて、変換規則格納部4から変換規則d.を検索する（ステップS105）。

【0059】

続いて、変換処理部2は、変換規則d.「タイプ<<Processing>>のクラスが定義するリソースノードを配置する。」を適用して、図7に示すように、タイプ<<Processing>>のリソースクラスC11が定義するリソースノードN01に変換して配置する（ステップS106）。

【0060】

次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し（ステップS107）、次のメッセージM03があるので、ステップS102に制御

を戻す。

【0061】

続いて、変換処理部2は、次のメッセージM03をカレントメッセージにセットして着目する（ステップS102）。

【0062】

次に、変換処理部2は、メッセージM03の送信元オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する（ステップS103）。

【0063】

詳しくは、変換処理部2は、まず、送信元オブジェクトOb03に着目し（ステップS108）、名前「Main」に基づいて送信元オブジェクトOb03を定義するシステムクラスC02に着目する（ステップS109）。次に、システムクラスC02に繋がるリソース関連線を探す（ステップS110）。リソース関連線L01が存在するので（ステップS111）、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプおよび属性として保存する（ステップS112）。

。

【0064】

続いて、変換処理部2は、メッセージM03の送信先オブジェクトOb04を定義するシステムクラスのタイプおよび属性を調査する（ステップS104）。

【0065】

詳しくは、変換処理部2は、まず、送信先オブジェクトOb04に着目し（ステップS108）、名前「A」に基づいて送信先オブジェクトOb04を定義するシステムクラスC03に着目する（ステップS109）。次に、システムクラスC03に繋がるリソース関連線を探す（ステップS110）。リソース関連線L02が存在するので（ステップS111）、リソース関連のあるリソースクラスC12のタイプ<<Storage>>および属性「評価対象外」をシステムクラスC03のタイプおよび属性として保存する（ステップS112）。

【0066】

次に、変換処理部2は、送信元オブジェクトOb03を定義するシステムクラス

C02のタイプ<<Processing>>および送信先オブジェクトOb04を定義するシステムクラスC03のタイプ<<Storage>>に基づいて、変換規則格納部4から変換規則e.を検索する(ステップS105)。

【0067】

続いて、変換処理部2は、変換規則e.「タイプ<<Storage>>のクラスが定義するリソースノードを配置する。」を適用して、図7に示すように、タイプ<<Storage>>のクラスC12が定義するリソースノード群N04, N05, N06に変換し、リソースクラスC12の属性が「評価対象」となっているので、ノード群N04, N05, N06を評価対象サブモデル60に配置する(ステップS106)。

【0068】

次に、変換処理部2は、シーケンス図51の次のメッセージの存在を調査し(ステップS107)、次のメッセージM04があるが、破線矢印のメッセージであるので、これを無視し、さらに次のメッセージの存在を調査し、メッセージM05があるので、ステップS102に制御を戻す。

【0069】

変換処理部2は、次のメッセージM05をカレントメッセージにセットして着目する(ステップS102)。

【0070】

次に、変換処理部2は、カレントメッセージM05の送信元オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する(ステップS103)。

【0071】

詳しくは、変換処理部2は、まず、送信元オブジェクトOb03に着目し(ステップS108)、名前「Main」に基づいて送信元オブジェクトOb03を定義するシステムクラスC02に着目する(ステップS109)。次に、システムクラスC02に繋がるリソース関連線を探す(ステップS110)。リソース関連線L01が存在するので(ステップS111)、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」を

システムクラス C 0 2 のタイプおよび属性として保存する（ステップ S 1 1 2）。

【 0 0 7 2 】

続いて、変換処理部 2 は、メッセージ M 0 5 の送信先オブジェクト O b 0 5 を定義するシステムクラスのタイプおよび属性を調査する（ステップ S 1 0 4）。

【 0 0 7 3 】

詳しくは、変換処理部 2 は、まず、送信先オブジェクト O b 0 5 に着目し（ステップ S 1 0 8）、名前「B」に基づいて送信先オブジェクト O b 0 5 を定義するシステムクラス C 0 4 に着目する（ステップ S 1 0 9）。次に、システムクラス C 0 4 に繋がるリソース関連線を探す（ステップ S 1 1 0）。リソース関連線 L 0 3 が存在するので（ステップ S 1 1 1）、リソースクラス C 1 3 のタイプ << S t o r a g e >> および属性「評価対象外」をシステムクラス C 0 4 のタイプおよび属性として保存する（ステップ S 1 1 2）。

【 0 0 7 4 】

次に、変換処理部 2 は、送信元オブジェクト O b 0 3 を定義するシステムクラス C 0 2 のタイプ << P r o c e s s i n g >> および送信先オブジェクト O b 0 5 を定義するシステムクラス C 0 4 のタイプ << S t o r a g e >> に基づいて、変換規則格納部 4 から変換規則 e . を検索する（ステップ S 1 0 5）。

【 0 0 7 5 】

続いて、変換処理部 2 は、変換規則 e . を適用しようとするが、システムクラス C 0 4 の属性が「評価対象外」となっているので、ノードへの変換を行わずに性能評価サブモデル 6 0 に配置しない（ステップ S 1 0 6）。

【 0 0 7 6 】

次に、変換処理部 2 は、シーケンス図 5 1 の次のメッセージの存在を調査し（ステップ S 1 0 7）、次のメッセージ M 0 6 があるので、ステップ S 1 0 2 に制御を戻す。

【 0 0 7 7 】

次に、変換処理部 2 は、シーケンス図 5 1 のカレントメッセージ M 0 6 に着目する（ステップ S 1 0 2）。

【0078】

続いて、変換処理部2は、メッセージM06の送信元オブジェクトOb03を定義するシステムクラスのタイプおよび属性を調査する（ステップS103）。

【0079】

詳しくは、変換処理部2は、まず、送信元オブジェクトOb03に着目し（ステップS108）、名前「Main」に基づいて送信元オブジェクトOb03を定義するシステムクラスC02に着目する（ステップS109）。次に、システムクラスC02に繋がるリソース関連線を探す（ステップS110）。リソース関連線L01が存在するので（ステップS111）、リソース関連のあるリソースクラスC11のタイプ<<Processing>>および属性「評価対象」をシステムクラスC02のタイプおよび属性として保存する（ステップS112）。

【0080】

次に、変換処理部2は、メッセージM06の送信先オブジェクトOb06を定義するシステムクラスのタイプおよび属性を調査する（ステップS104）。

【0081】

詳しくは、変換処理部2は、まず、送信先オブジェクトOb06に着目し（ステップS108）、名前「subP」に基づいて送信先オブジェクトOb06を定義するシステムクラスC05に着目する（ステップS109）。次に、システムクラスC05に繋がるリソース関連線を探す（ステップS110）。リソース関連線が存在しないので（ステップS111）、元のシステムクラスC05のタイプ<<Interface>>および属性をシステムクラスC05のタイプおよび属性として保存する（ステップS113）。

【0082】

次に、変換処理部2は、送信元オブジェクトOb03を定義するシステムクラスC03のタイプ<<Processing>>および送信先オブジェクトOb06を定義するシステムクラスC05のタイプ<<Interface>>に基づいて、変換規則格納部4から変換規則f.を検索する（ステップS105）。

【0083】

続いて、変換処理部 2 は、変換規則 f. を適用して、外部パッケージのタイプ<<Interface>>のクラスがメッセージを受け取る場合、外部パッケージに対する処理の依頼であるので、その外部パッケージに対応する性能評価サブモデル 6 0 のノード N 0 7 に変換し配置する（ステップ S 1 0 6）。このノード N 0 7 は、タイプ<<Interface>>のシステムクラス C 0 5 が属するパッケージ 5 0 を表している。

【0084】

次に、変換処理部 2 は、シーケンス図 5 1 の次のメッセージの存在を調査し、次のメッセージ M 0 7 があるが、破線矢印のメッセージであるので、これを無視し、さらに次のメッセージの存在を調査し、メッセージがないと判断する（ステップ S 1 0 7）。

【0085】

最後に、変換処理部 2 は、性能評価サブモデル 6 0 がシンクノードまたはリターンノードで完結していないので、変換規則 h. により、性能評価サブモデル 6 0 に終了処理を加える。この性能評価サブモデル 6 0 では、最初のオブジェクト O b 0 1 をソースノード N 0 2 としたので、シンクノード N 0 8 を配置し、性能評価サブモデル 6 0 を完結させる。

【0086】

なお、第 1 の実施の形態では、変換規則 a. ～ h. が変換規則格納部 4 に格納されているものとしたが、変換規則は変換規則編集部 3 により変換規則格納部 4 に追加および変更可能となっているので、上記に示した変換規則以外に従った変換も可能であることはいうまでもない。

【0087】

また、第 1 の実施の形態では、1 つのパッケージ 5 0 の性能評価サブモデル 6 0 について説明したが、この処理を繰り返すことにより複数の性能評価サブモデル 6 0 からなる性能評価モデル 6 の全体が同様にして得られることはいうまでもない。

【0088】

第 1 の実施の形態では、通常 of システム分析およびシステム設計で行われるよう

な機能的な観点からUMLモデル5を作成し、ハードウェアリソースを表すリソースクラスと、システムクラスとリソースクラスとの関連を示すリソース関連線とを追加することで、性能評価モデル6を生成することが可能となる。これにより、UMLモデル5の作成と性能評価モデル6の作成との距離が縮まり、最終的に、性能評価モデル6の作成時間および作成コストを軽減することが可能となる。

【0089】

(2) 第2の実施の形態

図8を参照すると、本発明の第2の実施の形態に係る性能評価モデル生成装置は、図1に示した第1の実施の形態に係る性能評価モデル生成装置に対して、性能評価モデル生成プログラムを記録した記録媒体100を備える点のみが異なっている。この記録媒体100は、磁気ディスク、半導体メモリ、その他の記録媒体であってよい。

【0090】

性能評価モデル生成プログラムは、記録媒体100からコンピュータでなる性能評価モデル生成装置に読み込まれ、UMLモデル解析部1、変換処理部2、変換規則編集部3、および変換規則格納部4として動作する。各部1～4の動作は、第1の実施の形態に係る性能評価モデル生成装置における対応する各部1～4と全く同様になるので、その詳しい説明を割愛する。

【0091】

【発明の効果】

第1の効果は、システムの分析および設計で広く使われているUMLモデルを元に、性能評価モデルを作成することにより、性能評価モデルの作成が容易になり、性能評価モデルの作成にかかる時間の削減、つまりコストの削減ができることである。

【0092】

その理由は、UMLモデルで純粹に機能モデルを作成する部分に関しては、これまでのシステム設計の手法をほぼそのまま流用でき、UMLモデルを元にしてハードウェアリソースとの対応を考えるだけで性能評価モデルが生成できるため

ある。

【0093】

第2の効果は、システムのUMLモデルに誤りが無ければ、性能評価モデルの作成時点で起こりうる性能評価モデルの作成誤りをなくすることが可能となることである。

【0094】

その理由は、UMLモデルに誤りがなく、そのUMLモデルから系統的に性能評価モデルを作成するため、性能評価モデルの作成時に起こりうる単純な誤り、勘違いなどを防ぐことが可能となるからである。

【0095】

第3の効果は、性能評価ツールの違いを越えて、既存のモデルや他の人が作成したモデルを広く参照することが容易になることである。

【0096】

その理由は、機能モデルとしてUMLという統一モデリング言語を使うことにより、異なる性能評価ツール間でも同一のUMLモデルを使ってシステムを表現できるためである。性能評価モデルは、一般的に、独自のフォーマットでモデルを記述する必要がある、このモデルは一般的に各ツールで異なっている。本発明によれば、UMLモデルという統一表記法でシステムの機能モデルを入力し、性能評価をするため、システムの機能モデルの入力も一般的なツールが使える、性能評価ツールの違いを越えて、既存資産の流用も容易となるからである。

【図面の簡単な説明】

【図1】

本発明の第1の実施の形態に係る性能評価モデル生成装置の構成を示すブロック図である。

【図2】

第1の実施の形態に係る性能評価モデル生成方法の概要を説明する図である。

【図3】

図2中のシーケンス図の内容を例示する図である。

【図4】

図 2 中のクラス図の内容を例示する図である。

【図 5】

図 2 中の性能評価サブモデルの内容を例示する。

【図 6】

第 1 の実施の形態に係る性能評価モデル生成方法の手順を示すフローチャートである。

【図 7】

リソースクラスに対応するノード群を例示する図である。

【図 8】

本発明の第 2 の実施の形態に係る性能評価モデル生成装置の構成を示すブロック図である。

【符号の説明】

- 1 UML モデル解析部
- 2 変換処理部
- 3 変換規則編集部
- 4 変換規則格納部
- 5 UML モデル
- 6 性能評価モデル
- 7 UML モデル入力ツール
- 8 性能評価ツール
- 50 パッケージ
- 51 シーケンス図
- 52 クラス図
- 60 性能評価サブモデル
- 61 動的パラメータ
- 100 記録媒体
- 311～316 結線
- C01～C05 システムクラス
- C11～C13 リソースクラス

L01～L03 リソース関連線

M01～M07 メッセージ

Ob01～Ob06 オブジェクト

S101 カレントメッセージセットステップ

S102 カレントメッセージ着目ステップ

S103 送信元オブジェクトのクラスタイプおよび属性調査ステップ

S104 送信先オブジェクトのクラスタイプおよび属性調査ステップ

S105 変換規則検索ステップ

S106 ノード変換・配置ステップ

S107 次メッセージ有無判定ステップ

S108 オブジェクト着目ステップ

S109 クラス着目ステップ

S110 リソース関連探索ステップ

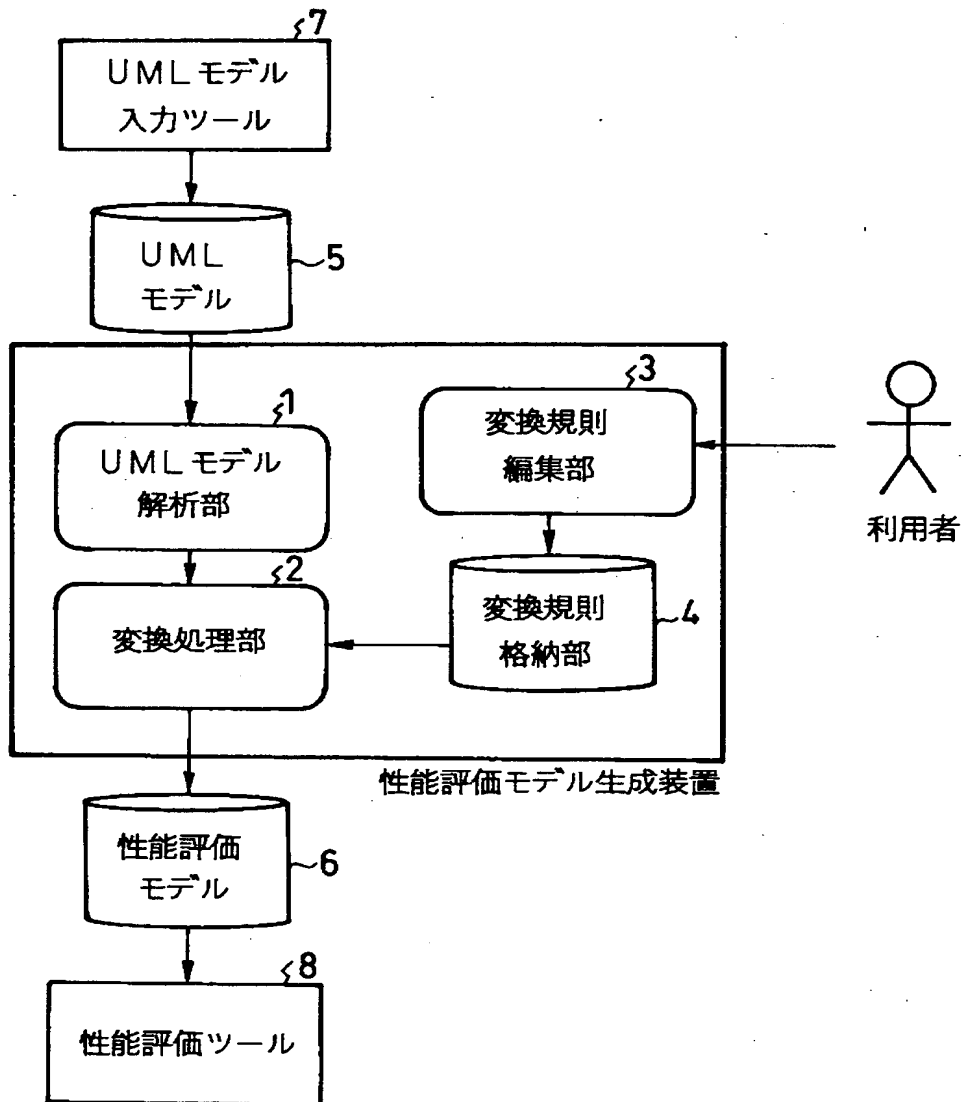
S111 リソース関連有無判定ステップ

S112 リソース関連先クラスタイプおよび属性保存ステップ

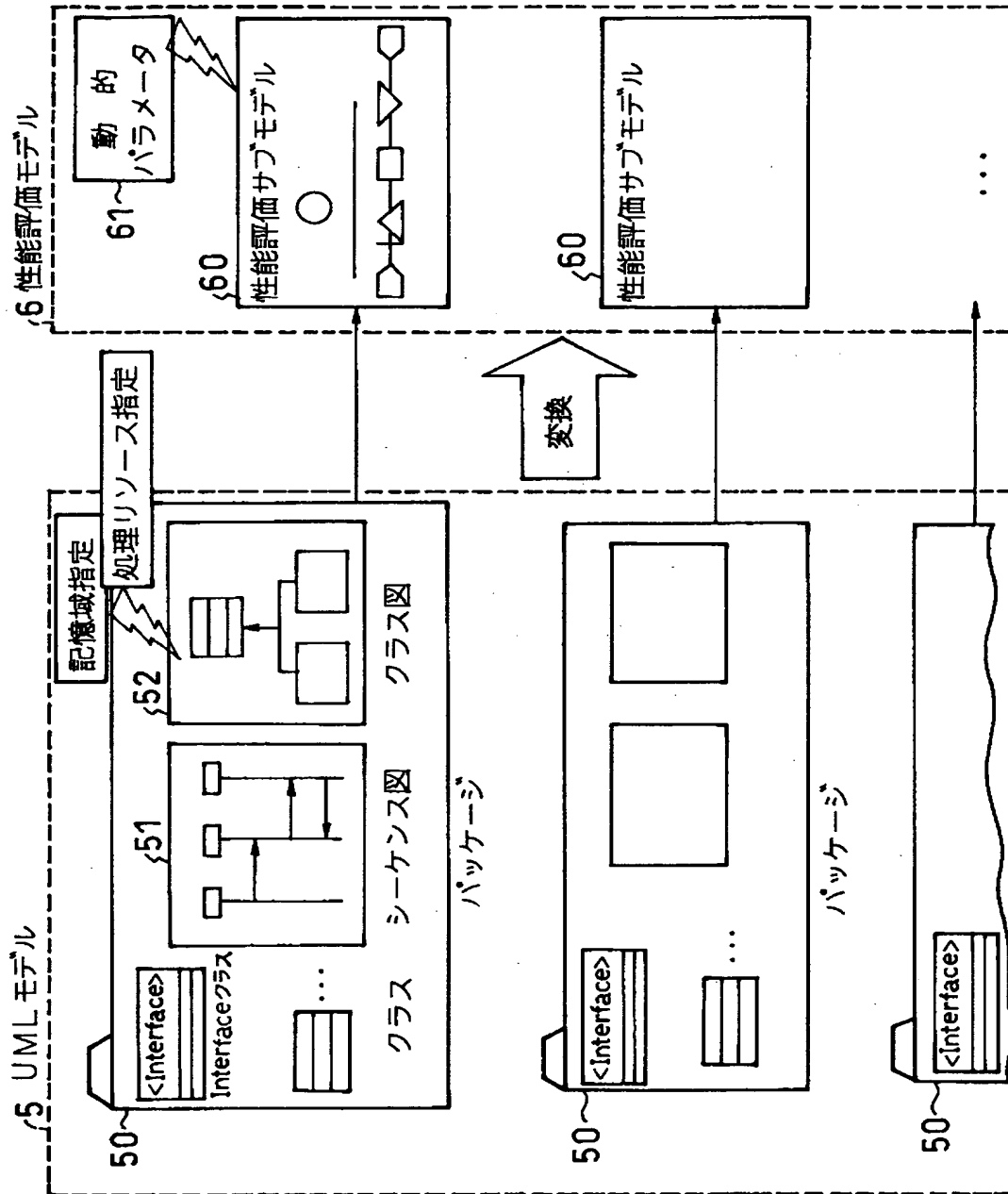
S113 自クラスタイプおよび属性保存ステップ

【書類名】 図面

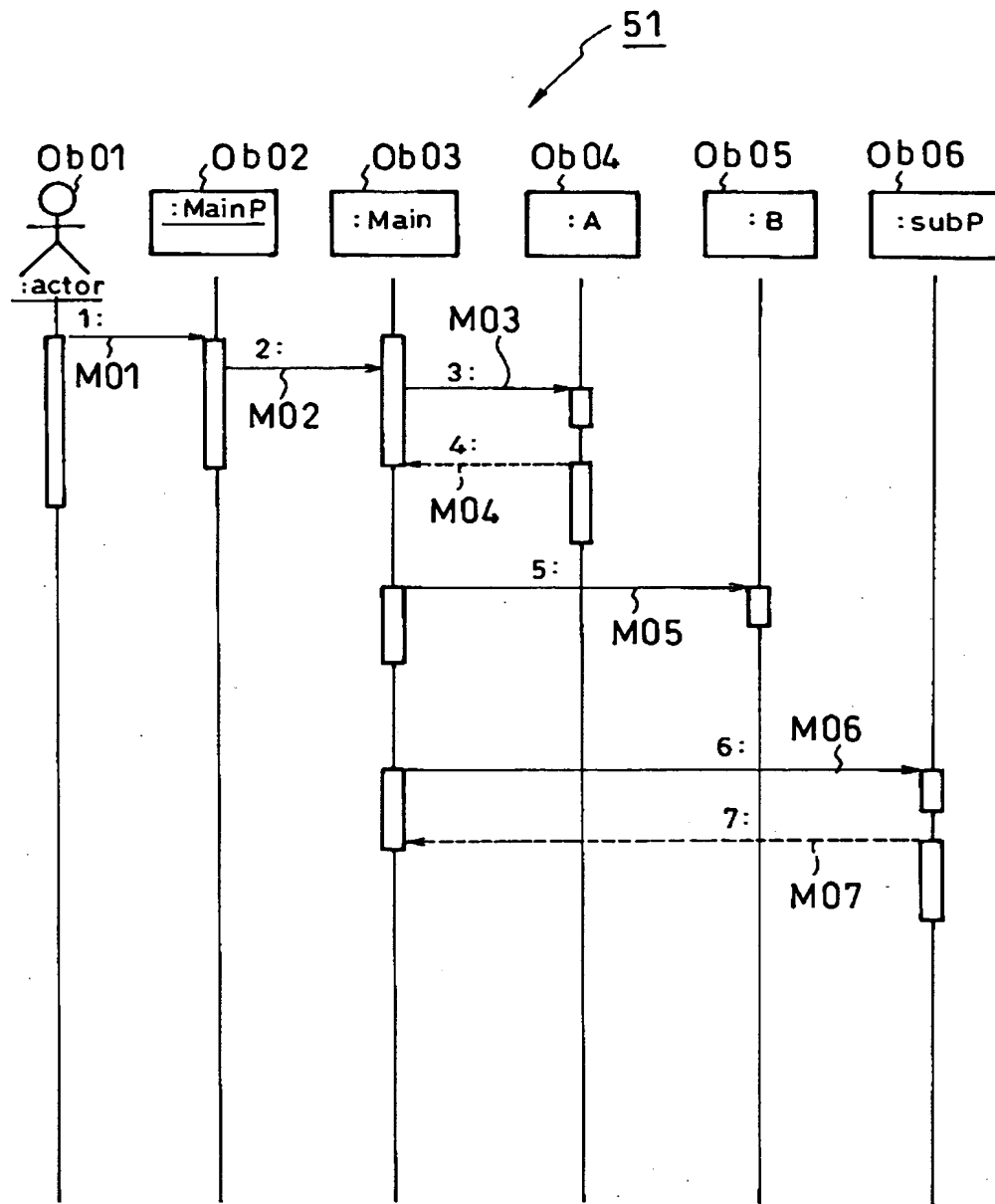
【図 1】



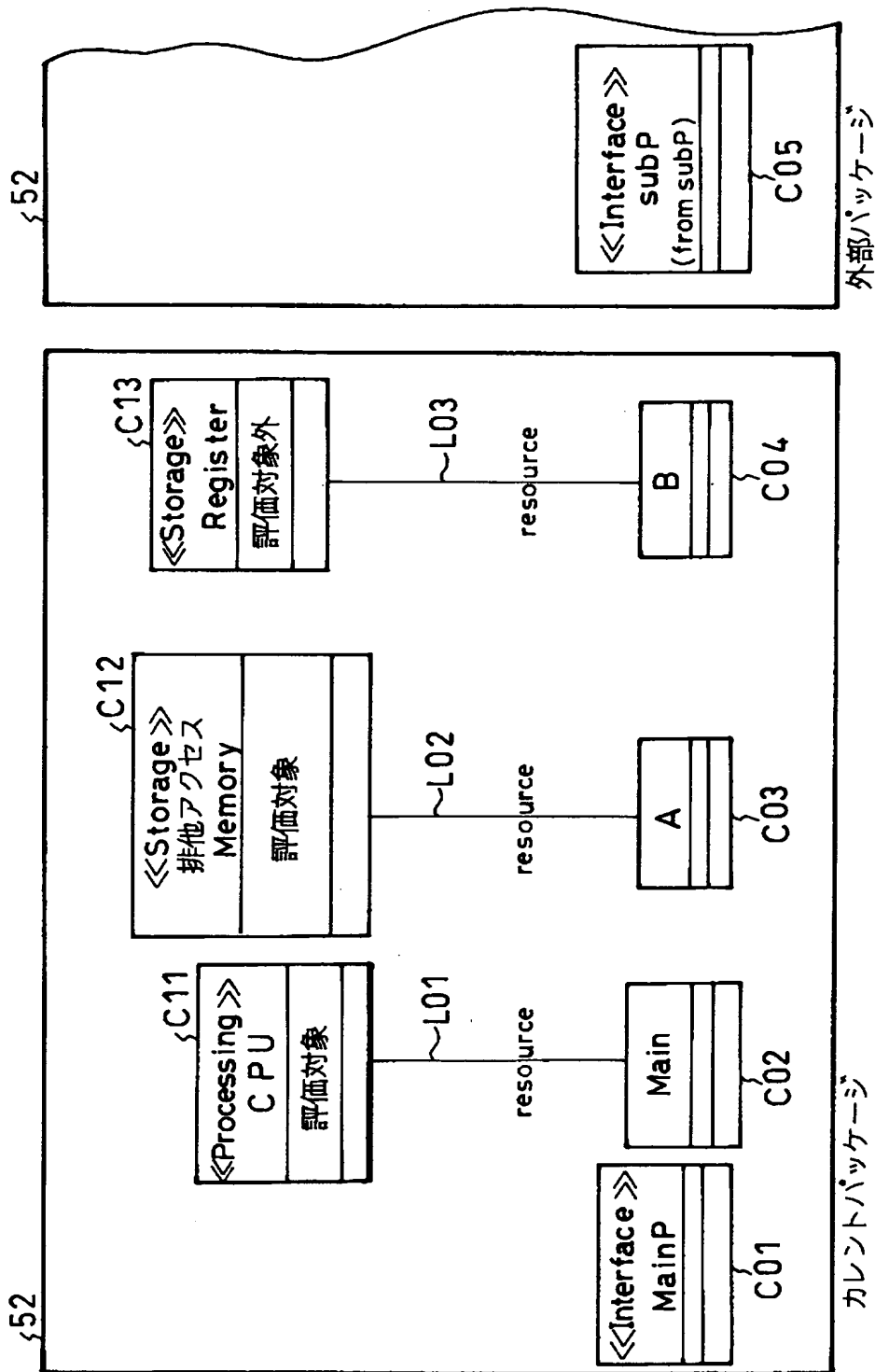
【図2】



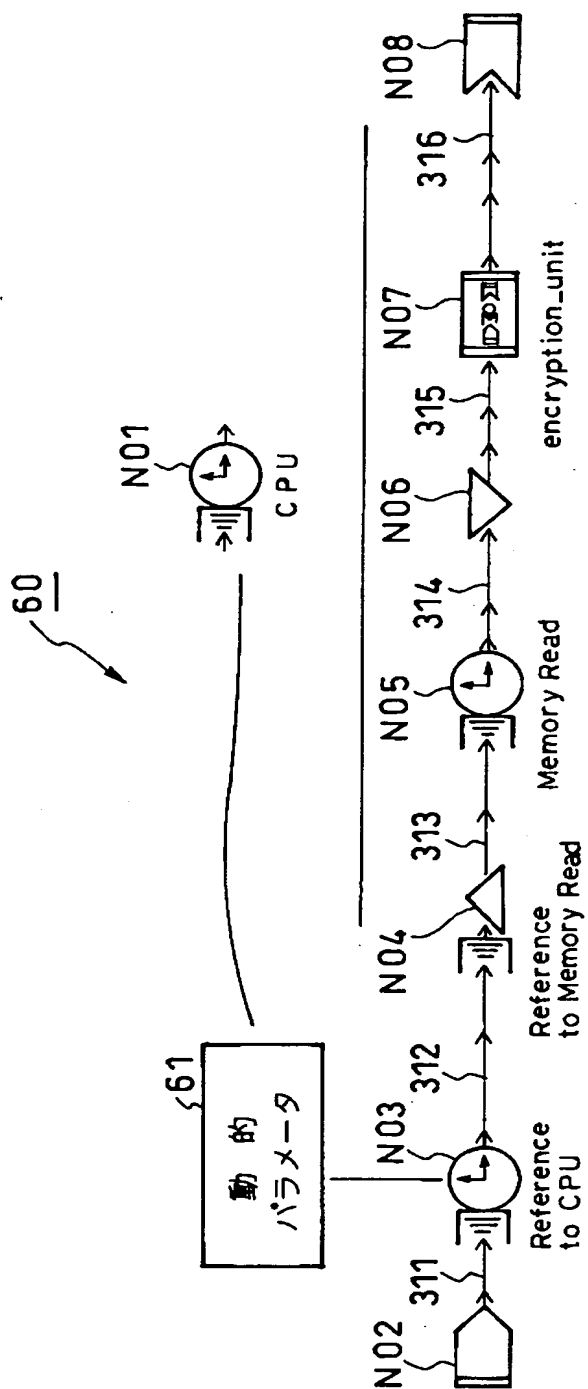
【図 3】



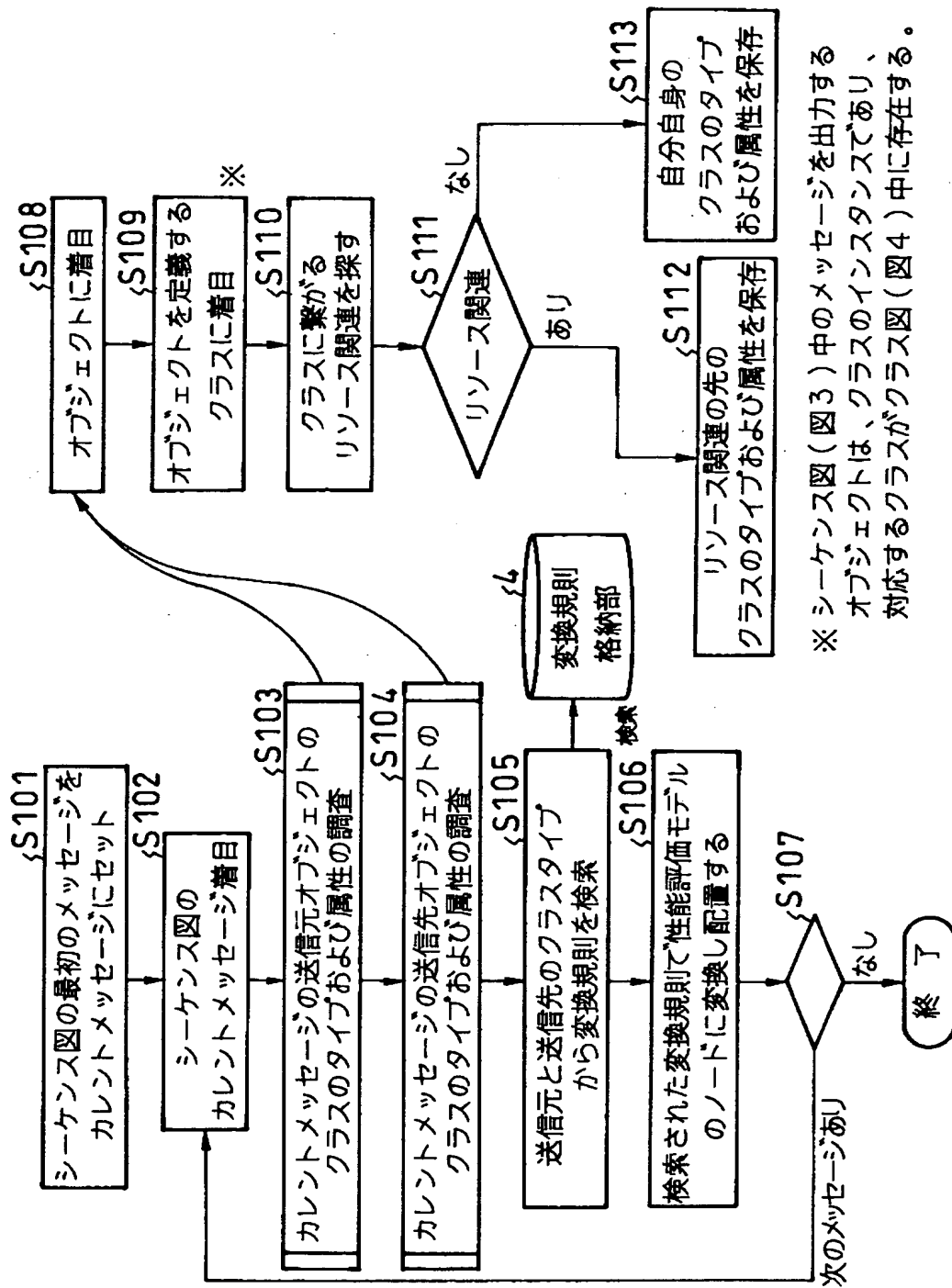
【図 4】



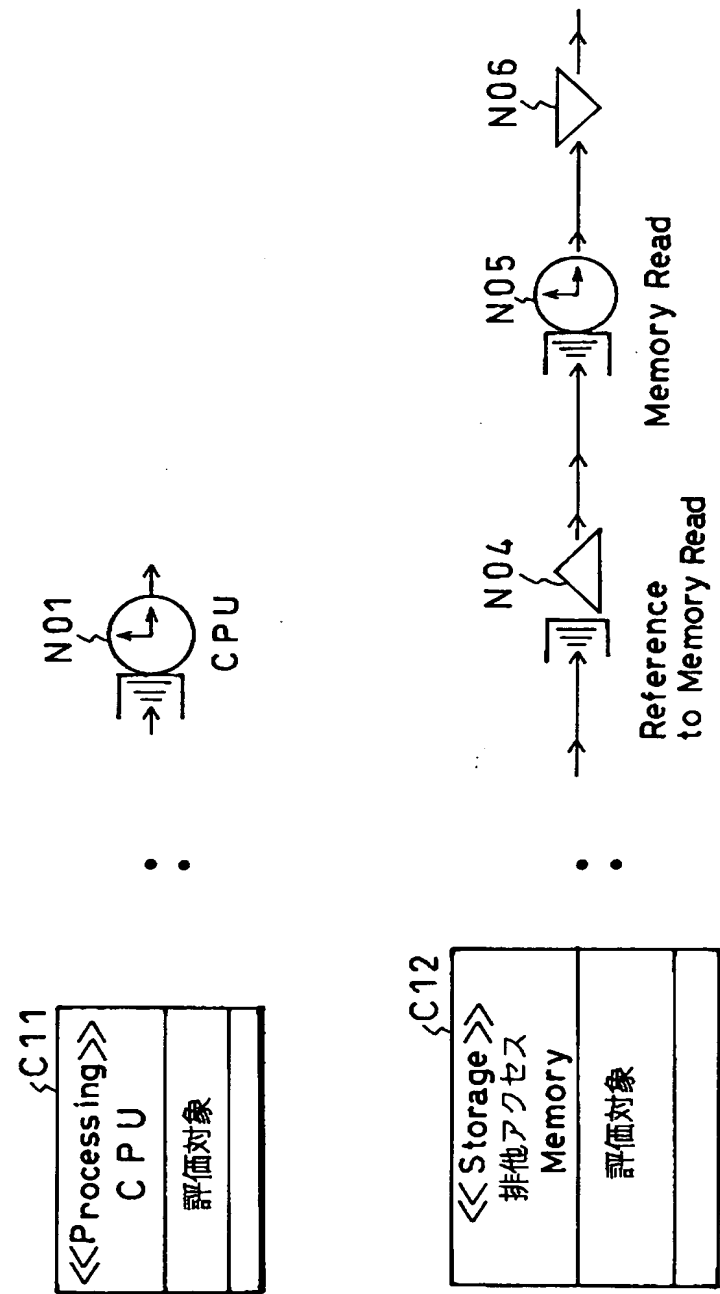
【図 5】



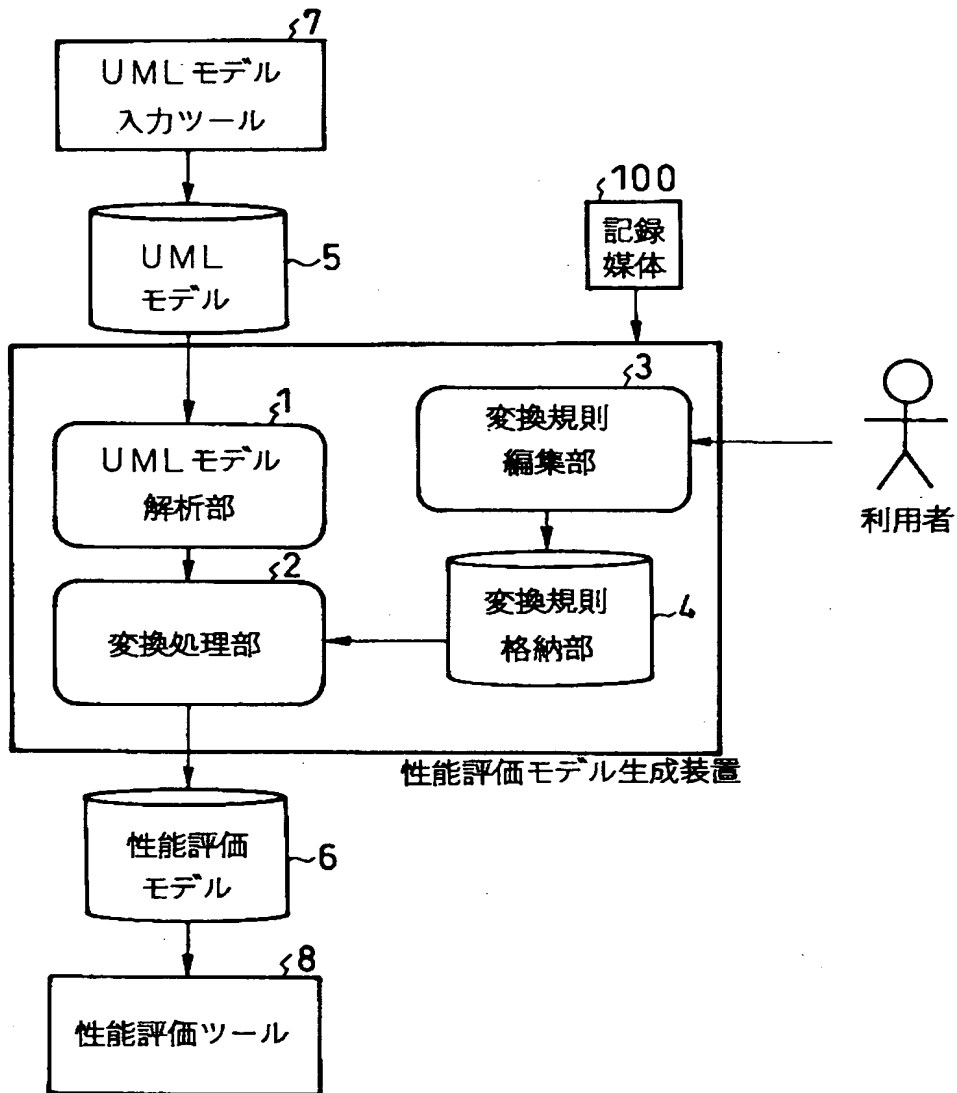
【図 6】



【図 7】



【図 8】



【書類名】 要約書

【要約】

【課題】 UMLモデルから性能評価モデルを自動的に生成できるようにし、性能評価モデルの作成にかかる時間およびコストを削減する。

【解決手段】 利用者は変換規則編集部 3 を使用して変換規則を入力し、変換規則格納部 4 に格納する。UMLモデル解析部 1 はUMLモデル 5 を入力して解析し、変換処理部 2 はUMLモデル解析部 1 による解析結果を変換規則格納部 4 の変換規則に従って変換して性能評価モデル 6 を生成する。

【選択図】 図 1

特2000-138391

認定・付加情報

特許出願の番号	特願2000-138391
受付番号	50000582065
書類名	特許願
担当官	第七担当上席 0096
作成日	平成12年 5月12日

<認定情報・付加情報>

【提出日】 平成12年 5月11日

次頁無

出 願 人 履 歴 情 報

識別番号 [000004237]

1. 変更年月日 1990年 8月29日

[変更理由] 新規登録

住 所 東京都港区芝五丁目7番1号

氏 名 日本電気株式会社